CHAPTER 1: INTRODUCTION TO DEBUG

The CROMEMCO DEBUG program makes it possible to test and debug user programs. DEBUG is loaded into memory and moved to the highest memory available below CDOS. When using a 32K CDOS and DEBUG, there is 20K left for the user program.

LOADING DEBUG

DEBUG is loaded by typing one of the following commands from CDOS.

```
DEBUG
DEBUG filename.ext
```

where "filename" is the name of the program to be tested, and "ext" is the file extension. In both cases, DEBUG is loaded into memory directly below CDOS. The CDOS jump instruction located at location 5H is changed to jump to the start of DEBUG. This allows locations 6H and 7H to still point to the lowest available memory location.

The second command above is used to load the file to be tested into memory. If the extension ("ext") is ".HEX", then the file is read as an INTEL HEX file. Any other extension is read as an absolute binary file, loaded at location 100H. Note that DEBUG does not load relocatable files. If an extension is ".REL" it will be loaded in as if it were binary and will not be executable.

CONTROL CHARACTERS

Control characters are used in DEBUG and TRACE to help in entering commands. These control characters are the same as CDOS uses.

Control-C (^C)	go back to CDOS
Control-H (^H)	delete character and backspace on CRT
Control-U (^U)	delete line
Control-X (^X)	delete character and echo
underscore	delete character and backspace on CRT
RUBout (DEL)	delete character and backspace on CRT

During a printing (such as from the DM command) the following characters may be used.

Control-S (^S)	stop/start printing. If printing, this character will stop the printing. If already stopped, this character will resume the printing.
break	(or any other character) will abort the printing, prompt, and wait for the next command.

COMMAND FORMAT

DEBUG is controlled by one and two character commands from the terminal. The format is free-form with respect to spaces. Commas may be used in place of spaces. In the following, the examples all dump memory starting at location 1000H and ending at location 10FFH.

```
DM1000 10FF (CR)
DM1000S100 (CR)
D M 1000 10FF (CR)
D M 1000 S 100 (CR)
DM1000,10FF (CR)
DM1000,S100 (CR)
D M 1000 , 10FF (CR)
```

@ REGISTER

DEBUG was designed to give flexibility in testing relocatable programs. The "@" register is used to tell DEBUG where the module you wish to debug is located. This address can be found from the map generated by the linking loader "LINK". To change the "@" register, type "@ (CR)" on the console. The computer will then type "@-xxxx " (where xxxx is the current value of the register). The computer will then wait for a new address. If a CR only is typed, the register remains unchanged. If an address and a CR is typed, then the register will contain the new address. The "@" register may now be used as part of an address. The following example demonstrates its use.

```
G/@ @A3 1000
```

This is an example of the go command. Break points will be set at the beginning of the current module, relative location A3H in the current module, and at location 1000H. This feature allows you to test a module without having to calculate absolute addresses.

ADDRESS EXPRESSIONS

For additional ease in specifying addresses an expression can be used. Within these expressions, addition, subtraction, the "@" register, and the "$" may be used. The "$" is the current location of the program counter (P register). If many modules are being tested, addition can be used to specify relative addresses.

```
G/2321+A3
```

The preceding example would set a break point at relative location A3H if the module is located at 2321H.

SWATH OPERATOR

There are two ways to specify the address range of many commands. The first is to simply list the beginning and end addresses (and where appropriate, the destination address). For example, the first command below programs the range Ø through 13FFH into PROMs starting at location E4ØØH. The second command displays the contents of memory between addresses E4ØØH and E4Ø2H.

```
PØ 13FF E4ØØ
DME4ØØ E4Ø2
```

Another way to do the same thing is to use the Swath operator, "S", to specify the width of the address range, rather than state the end address explicitly.

```
PØ S14ØØ E4ØØ
DM E4ØØS3
```

ERRORS

Any errors made during entering of a command may be corrected by typing Control-U (^U) to abort the line or by backspacing and correcting the line. If a CR has already been entered and DEBUG detects an error, the line will not be accepted and a "?" will be printed. Re-enter the line with the incorrect data corrected.

CHAPTER 2: DEBUG COMMANDS

DEBUG and TRACE commands are described in detail below. The operator must wait for the prompt character ("-") before entering the command.

A - Assemble into memory

This command allows in-line assembly language to be assembled into memory. The command takes the following format.

A beginning-addr (CR)

The user is prompted with the absolute address, followed by the relative address. DEBUG reads from the console the assembler mnemonics and assembles the instruction into memory. The mnemonics for the various Z-80 instructions can be found in the Z-80 CPU TECHNICAL MANUAL published by Mostek and Zilog. If there was no error in the instruction, it is stored in memory and the user is prompted for the next instruction. The rules for address expressions apply to the addresses in the assembler mnemonics. In the following example the "@" register contains 1234H.

```
A@40
1274 0040' ADD B
1275 0041' CALL @93
1278 0044' JP 1032+95
127B 0047' .
```

The A command terminates when the first blank line or a line starting with a "." is entered from the console. If there is an error in the input line, it will not be accepted, a "?" will be printed and the console will be prompted with the addresses again.

DM - DISPLAY MEMORY

The contents of memory are displayed in hexadecimal form. Each line of the display is preceded by the address of the first byte and followed by the ASCII representation of the hexadecimal bytes. An example follows:

```
DM100,S30
0100  40 41 42 43 44 45 46 47-48 49 4A 4B 4C 4D 4E 4F  @ABCDEFGHIJKLMNO
0110  50 51 52 53 54 55 56 57-58 59 5A 30 31 32 33 34  PQRSTUVWXYZ01234
0120  35 36 37 38 39 00 00 00-00 00 00 00 00 00 00 00  56789...........
```

The formats of this command are as follows.

```
DM (CR)
DM beginnig-addr (CR)
DM beginning-addr ending-addr (CR)
DM beginning-addr S swath-width (CR)
DM,ending-addr (CR)
DM S swath-width (CR)
```

The first format displays memory from the CURRENT display address, initially 100H, and continues for 8 lines. The second format displays from the beginning address and continues for 8 lines. The third format displays from the beginning address to the ending address. The fourth format displays from the beginning address for a length specified by the swath-width. The fifth format displays from the CURRENT display address to the ending address. The sixth format displays from the CURRENT display address for a length specified by the swath-width.

If an "X" is included after the "DM", the relative addresses are also printed. In the following example assume that the "@" register contains 100H.

```
DMX100,S30
0100 0000' 40 41 42 43 44 45 46 47-48 49 4A 4B 4C 4D 4E 4F  @ABCDEFGHIJKLMNO
0110 0010' 50 51 52 53 54 55 56 57-58 59 5A 30 31 32 33 34  PQRSTUVWXYZ01234
0120 0020' 35 36 37 38 39 00 00 00-00 00 00 00 00 00 00 00  56789...........
```

DR - DISPLAY REGISTERS

When DEBUG is re-entered from a break point, the user registers are saved. The registers may be displayed at any time by typing the following command.

```
-DR (CR)
SZHVNCE A=00 BC=0000 DE=0000 HL=0000 S=0100 P=0100 0100' LD   E,A
SZHVNC A'=00 B'=0000 D'=0000 H'=0000 X=0000 Y=0000 I=00
```

The letters "SZHVNC" on the first row represent the flags, while on the second row they represent the prime flags. If the flag is on, it is printed, if the flag is off, a space is printed. If only the carry and zero flag are set then " Z C" would be printed. The flags are described below.

S - Sign flag, S=1 if the MSB of the result is one, i.e., the result is negative.

Z - Zero flag, Z=1 if the result of an operation is zero.

H - Half-carry flag, H=1 if the add operation produced a carry into the 4th bit of the accumulator or a subtract operation produced a borrow from the 4th bit of the accumulator.

V - Parity or overflow flag. This flag is affected by arithmetic and logical operations. If an overflow occurs during an arithmetic operation, the flag is set to one. After a logical operation, the flag is set to 1 if the result of the operation has even parity.

N - Add/subtract flag, N=1 if the last operation was a subtraction.

C - Carry flag, C=1 if the operation produced a carry.

The E flag on the first line is the state of the interrupt enabled flip-flop (IFF). If interrupts are enabled, the "E" is printed, otherwise a space is printed.

The A register is printed next, followed by the BC, DE, and HL register pairs and the stack pointer. The program counter value is then printed in both absolute and relative. The opcode pointed to by the program counter is then displayed as an instruction.

On the second line, the prime registers are displayed, F' (prime flags), A', BC', DE', and HL'. The IX, IY, and I (interrrupt page) registers are printed next. If the disassembled opcode includes an address, the relative value of this address is printed as the last thing on the line.

```
-DR (CR)
S H NCE A=00 BC=0000 DE=0000 HL=0000 S=0000 P=1234 0010' CALL 1334
SZ NC A'=00 B'=0000 D'=0000 H'=0000 X=0000 Y=0000 I=00       (0110')
```

E - EXAMINE INPUT PORT

The data port is read and displayed as a hexadecimal number. The format of the command is:

E data-port (CR)

In the following example the data port 3 is read and displayed on the console.

```
-E3 (CR)
23
```

EJ - EJECT DISK

The format of the command follows.

EJ d

Where d is the disk number (A, B, C, D). If the designated disk is a CROMEMCO DUAL DISK SYSTEM model PFD, with the eject option, the diskette in the disk drive will eject.

F - SPECIFY FILE NAME

This command allows the operator to insert filenames in the two default FCBs (at 5CH and 6CH) and the command line into the default buffer (at

80H). The example below loads FILE1.COM into the first FCB and FILE2.COM into the second FCB. The complete line is also loaded into the default buffer.

```
-FFILE1.COM FILE2.COM OPTION1 OPTION2
```

This command can be used with the "R" command to read in disk files.

G - GO

The GO command has the following format.

```
G(starting-addr)/(breakpoint-1) (breakpoint-2)...(breakpoint-5)
```

Each of the addresses is optional. If the starting address is omitted, then the contents of the program counter is used. The registers are loaded from the user registers (these are the values displayed with the DR command). Execution begins with the starting address or the contents of the program counter. If break points were specified, an RST 30H is inserted at the break point addresses and a jump instruction is placed at location 30H. When a breakpoint is executed, control is returned to DEBUG, and all of the user registers are saved (the registers may then be displayed with the DR command). ALL breakpoints are then removed from the user program. The program counter is displayed after the breakpoint. Note the following about breakpoints:

(a) Breakpoints can only be set in programs residing in RAM. This is because an RST 30H is inserted at each break point location. (The original contents of these locations are saved so that they can be restored after a break point is executed.)

(b) Up to 5 break points can be set. If an attempt is made to enter more than 5 break points, the command will not be accepted.

(c) When a break point is used, a jump instruction is stored at location 30H. Therefore locations 30H, 31H, and 32H are not available to a user program.

The GO command has an additional feature that

is very helpful in debugging a program. A count is allowed for each break-point. This count is entered after the break-point and enclosed in parentheses. This count is the number of times the program reaches this address before control is returned to DEBUG. A count of one says to break the next time the address is reached. In the example below execution begins at location 100H and will break when address 109H is reached for the second time or when 123H is reached for the first time.

```
-G100/109(2) 123
```

Note that 123 and 123(1) means the same thing. Also note that the count is a hexadecimal number. Therefore 123(F) means to break after the address has been executed for the 15th time.

H - HEXADECIMAL ARITHMETIC

Hexadecimal addition and subtraction may be performed by this command. The first number to be printed is the sum of the two input numbers. The second number to be printed is the difference between the first number and the second number. In the example following, the first number is 1234 + 321, and the second number is 1234 -321.

```
-H1234,321
1555 0F13
```

L - LIST IN ASSEMBLER MNEMONICS

The list command is used to list the contents of memory in assembly language mnemonics. The formats for this command are.

```
L (CR)
L starting-addr (CR)
L starting-addr ending-addr (CR)
L starting-addf S swath-width (CR)
L,ending-addr (CR)
L S swath-width (CR)
```

The first format lists 16 lines of disassembled code starting from the current list address. The second format lists 16 lines from the starting address. The third format lists from the

starting address to the ending address. The fourth format lists from the starting address for a length specified by the swath width. The fifth format lists from the current list address to the ending address. The sixth format lists from the current address for a length specified by the swath address.

The first address of the disassembly is the absolute address. The second address is the relative address. If the disassembled instruction contains an address, the absolute address is printed in the instruction in hexadecimal and the relative address is printed to the right of the disassembled line. In the example that follows, the "@" register contains 2800H.

```
-L@800 812
3000 0800' ADD B
3001 0801' CALL 3200            (0A00')
3004 0804' CALL 3243            (0A43')
3007 0807' CALL 3333            (0B33')
300A 080A' LD A,B
300B 080B' OR C
300C 080C' JR Z,3000            (0800')
300F 080F' INC HL
3010 0810' INC DE
3011 0811' INC BC
3012 0812' LD A,H
```

M - MOVE MEMORY

The formats of this command follow.

```
M source-addr source-end destination-addr
M source-addr S swath-width destination-addr
```

The first format moves the contents of memory beginning with the source address and ending with the source-end to the destination address. The second format uses the swath width to determine the length of the move.

The move is verified to insure that all bytes were moved correctly. If an overlapping move was made, errors will be reported. The error reporting can be terminated by typing any character.

The move command can be used to fill a block of memory with a constant. In the following

example, a zero has been entered into location 100H using the SM command. The following command will move zeros from location 100H through 108H.

```
-M100 S7 101
```

Care should be taken not to move memory over DEBUG, TRACE or CDOS.

O - OUTPUT TO DATA PORT

This command outputs data to a data port. The following is the command format.

```
O data-byte port-number (CR)
```

P - PROGRAM PROMS

This command allows programming of PROMs. The following are the command formats.

```
P source-addr source-end destination-addr
P source-addr S swath-width destination-addr
```

The first format programs PROMs starting with the source address and ending with the source-end into PROMs beginning at the destination address. The second format determines the length from the swath width.

If the length of the source is not a multiple of 400H or if the destination does not begin at a 400H boundary, DEBUG will reject the command. (Multiples of 400H end in '000', '400', '800', and 'C00'.)

Any number of 2708 or 2704 PROMs can be programmed in the execution of one command as long as there are enough BYTESAVERS to contain them. Each PROM is verified with its source after all are programmed and any discrepancies are printed out. If no discrepancies are found, a prompt is printed and the next command may be entered.

Software can be loaded into a PROM in as small increments as you desire, provided it is added to previously unused areas of the PROM. This is done by first using the Move command, "M", to transfer the contents of the PROM to RAM, adding the new

SM - SUBSTITUTE MEMORY

This command is used to substitute memory. The format of the command follows.

SM starting-addr

DEBUG prints the absolute address, followed by the relative address, followed by the contents of the memory byte. One of the following may then be entered.

(a) data-byte value. The data byte value is stored at the address of the prompt. The address is then incremented by 1 and displayed on the next line.

(b) string enclosed in quotes. The string is stored beginning at the address of the prompt. The address is then incremented past the string and displayed on the next line.

(c) Any number of (a) and (b) above can be entered on one line. The address is then incremented past the bytes that were stored and the new address is displayed on the next line.

(d) "-". A minus sign does not store a byte. The address will be decremented to the previous address. The minus sign can be used to "back up" to a previous location in case an error has been made.

(e) (CR) only. If no entry is made on the line, the memory byte remains unchanged. The address is incremented by 1 and displayed on the next line.

(f) period. A period ends the input mode and returns to the command level.

In the example that follows, assume that the "@" register contains the value 2800H.

```
-SM@100
```

```
2900 0100' 32 0
2901 0101' 17 00
2902 0102' 31 'THIS IS AN ASCII STRING'
2919 0119' 7A 'AAAA' 0 0 1 2 3 4 5 6 7 8 9
2928 0128' 22
2929 0129' 29
292A 012A' 87 -
2929 0129' 29 .
```

Sr - SUBSTITUTE REGISTER

The Sr command allows the user registers to be altered. The letter "r" stands for the register which is to be changed. The section SUMMARY OF REGISTER NAMES gives a summary of the names that can be substituted. When substituting the F and F' flags, enter the command SF or SF'. DEBUG will then print the flags that are set and wait for the operator to enter the names of the registers that are to be set. If the flags are NOT entered, the flags are reset. In the following example, the "SZHC" flags are set. After the example is executed the "ZC" flags are set. The lower case letters are entered by the operator.

```
-sf
SZH C zc
```

When sustituting a one byte register, a one byte value is accepted. When substituting a two byte register, a two byte value is accepted. If no value is entered, or if an error occurs, the value of the register remains unchanged. In the following example, the A register is changed to contain 41H.

```
-sa
A=98 41
```

T - TRACE

The format of trace is:

```
T (CR)
T number-of-lines (CR)
```

The first format traces the program through one instruction. The second format traces the program through "number-of-lines" instructions. After every instruction traced, the values of the user registers are printed in the same format as the "DR" command.

You can trace only through RAM. The trace command places a break point after the instruction, loads the registers and executes the instruction. The break point is then executed and the registers are resaved. The registers are printed, and the next instruction is executed unless the count has reached zero, in which case a prompt is printed and you may enter the next command.

To abort the trace, hit any key on the console. A prompt will be printed and you may enter the next command.

TN - TRACE WITH NO PRINTING

The "TN" command is the same as the "T" command with the exception that after every instruction is traced, the registers are not printed. Only the last traced instruction is printed.

V - VERIFY MEMORY

Verify that the block of memory between souce address and source end contain the same value as the block beginning at destination address. The addresses and contents are printed for each discrepancy found. The following is the format of this command.

```
V source-addr source-end destination-addr
V source-addr S swath-width destination-addr
```

This command works by reading bytes from the source and destination and comparing them. If a discrepancy is found, the memory is read again for print-out. Thus, it can happen that a discrepancy is printed-out with the source and the destination contents indicated to be the same. This is caused by a defective memory element.

A discrepancy is printed in the following order, source address, source contents, destination contents, destination address. In the example that follows, memory locations 1003H and 1008H are defective.

```
-V Ø S3Ø 1ØØØ
ØØØ3 32 12 1ØØ3
ØØØ8 7A 5A 1ØØ8
-
```